RYERSON POLYTECHNIC UNIVERSITY

DEPARTMENT OF MATH, PHYSICS, AND COMPUTER SCIENCE

CPS 710

FINAL EXAM

FALL 94

STUDENT ID: _____

INSTRUCTIONS

Please write your student ID on this page. Do not write it or your name on any other pages.

Please answer directly on this exam.

This exam has 4 questions and is worth 30% of the course mark.

CALCULATORS ARE ALLOWED. NO OTHER AIDS ARE ALLOWED.

A - General Concepts	35
B - Shift-Reduce Parsing	15
C - Evaluation	25
D - Grammars	25

This exam has 7 pages including the front page

This exam has 7 pages including the front page

Part A - General Concepts - 35 points

A1 (6 points)

Give **2** reasons why you would want to have the scanning module of an interpreter separate from the rest of the program.

A2 (4 points)

What would be the output of the following program fragment if the language used was **dynamically scoped**?

```
integer a = 1;
integer b = 2;
function f( integer a)
    {
    b := a + 5;
    return b;
    }
function g(integer a)
    {
    integer b;
    return f(a);
    }
display g(6);
display b;
```

A3 (4 points)

What would be the output of the program fragment in A2 if the language was **statically scoped**?

A4 (5 points)

List 5 different types of semantic errors in programs

A5 (6 points)

What is an activation record? List 2 fields of an activation record.

A6 (5 points)

Explain why an interpreter needs to manage its heap space, and describe a memory management method for doing so.

A7 (5 points)

Give the definition of a rightmost derivation. You may use an example if necessary.

Part B - Shift-Reduce Parsing (15 points)

Given the LR(1) grammar with the following productions augmented with states (shown subscripted and outlined):

(1) FNCALL $\varnothing_1$ **f** $_2$ **(** $_3$ PARAMS $_8$ **)** $_9$

(2) PARAMS $\varnothing_3$ ε $_3$

(3) PARAMS $\varnothing_3$ PARLIST $_5$

(4) PARLIST $\varnothing_3$ **a** $_4$

(5) PARLIST $\varnothing_3$ PARLIST $_5$ **,** $_6$ **a** $_7$

Fill in the LR table for this grammar:

	FNCALL	PARAMS	PARLIST	**f**	**(**	**)**	**,**	**a**	**$**
1									
2									
3									
4									
5									
6									
7									
8									
9									

There are 4 possible entries for each slot in the table:

- S state: shift token onto symbol stack and change state
- R production: reduce the production by popping n symbols off the symbol stack and n symbols off the state stack, and inserting the left-hand side of the production into the input stream
- HALT: to halt process
- nothing: to indicate a syntax error

Part C - Evaluation - 25 points - This has short answers

In this question you will be evaluating **cond** expressions in Scheme.

- The section of the grammar which deals with **cond** expressions is:
  COND ∅ **(** **cond** CLAUSES **)**
  CLAUSES ∅ **(** PREDICATE CONSEQUENT **)** {CLAUSES}^{0}
  CLAUSES ∅ **(** **else** CONSEQUENT **)**
  The non-terminals PREDICATE and CONSEQUENT are functional expressions defined elsewhere in the grammar. You may assume that they are properly parsed and evaluated.
- A syntax tree node produced by the parser has the structure:

```
typedef struct node {
        int type;
        struct node *v1, *v2;
        } node;
```

- a COND structure is organised as follows:
  - type = COND
  - v1 points to the first clause in the COND expression
  - v2 points to the other clauses in the COND expression
- a CLAUSE structure is organised as follows
  - type = CLAUSE
  - v1 points to the predicate of the clause.
  - v2 points to the consequent of the clause.
- You are writing int *eval(node *tree), a function which evaluates a syntax tree. Eval returns a pointer to a structure containing the value of the syntax tree. Eval is completely written except for the part which handles COND and CLAUSE structures.
- The evaluation rules for COND and CLAUSE are:
  CLAUSE: if the predicate evaluates to TRUE (a special kind of pointer), then the value of the clause is the value of the consequent. Otherwise it is NULL (another special kind of pointer). The value of an else clause is the value of the consequent.
  COND: the value of the COND is the value of the first clause which doesn't evaluate to NULL.

C1 (5 marks)

What is in the v1 field of an else clause?

C2 (20 marks)

- Write C code for the section of eval which handles COND and CLAUSE structures. Indicate the context of this code.
- You do not need to do any memory management in your code. You can also assume that the expressions you are evaluating are error-free.

Part D - Grammars - 25 points

In this question, non-terminals are in UPPER-CASE and terminals are underlined**.**

D1 (6 points)

Left-factor the following set of productions fully. You may need to introduce new non-terminals.

SET ∅ SETOP | SIMPLE_SET

SETOP ∅ SET ≋ SET

SETOP ∅ SET ↔ SET

SIMPLE_SET ∅ □ | {1} | {2}

D2 (7 points)

Remove the **right recursion** from the following set of productions. You can make the grammar left-recursive if you wish.

S ∅ E { ± S }^{0}

E ∅ T { * E }^{0}

E ∅ (S)

T ∅ 0 | 1 | 2

D3 (6 points)

What is an **LR(1)** grammar? Why are LR(1) grammars useful?

D4 (6 points)

Why is the following set of productions **ambiguous**?

POLYN ∅ { + | − }^{0} COEFFICIENT x POWER POLYN_TAIL^0

POLYN_TAIL ∅ { + | − }^{0} COEFFICIENT x POWER POLYN_TAIL^0

COEFFICIENT ∅ { + | − }^{0} INTEGER | ε

POWER ∅ INTEGER | ε

INTEGER ∅ { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }$^+$